**1.Understand Sorting Algorithms**

a)Bubble Sort

Concept:  
Bubble Sort repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process is repeated until the list is sorted.

Example Behavior:  
Imagine bubbles rising to the top; heavier ones keep sinking—hence, large values "bubble" to the end.

b)Insertion Sort

Concept:  
Insertion Sort builds the sorted list one item at a time by comparing the current element to those before it and inserting it into its correct position.

Example Behavior:  
Like sorting playing cards in your hand—one by one, you insert the next card into its correct position.

c)Merge Sort

Concept:  
Merge Sort is a divide and conquer algorithm. It splits the list into halves, recursively sorts them, and merges them back in sorted order.

Example Behavior:  
Splitting a deck of cards in half, sorting each half, and merging them into one sorted deck.

d)Quick Sort

Concept:  
Quick Sort also uses divide and conquer. It selects a “pivot” element and partitions the array such that elements less than the pivot go left, greater go right. It then recursively sorts the partitions.

Example Behavior:  
Like organizing a group of people around a “middle” person based on height—shorter to one side, taller to the other.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Best** | **Average** | **Worst** | **Stable** | **Space** |
| Bubble Sort | O(n) | O(n2) | O(n2) | Yes | O(1) |
| Insertion Sort | O(n) | O(n2) | O(n2) | Yes | O(1) |
| Merge Sort | O(n log n) | O(n log n) | 0(n log n) | Yes | O(n) |
| Quick Sort | O(n log n) | O(n log n) | O(n2) | No | O(log n) |

**4.Analysis**

Time Complexity Comparison

|  |  |  |
| --- | --- | --- |
| **Metric** | **Bubble Sort** | **Quick Sort** |
| Best Case | O(n); Already Sorted | O(n log n) |
| Average Case | O(n2) | O(n log n) |
| Worst Case | O(1) | O(n2) |
| Space Complexity | O(1) | O(log n) |

Quick Sort is preferred over Bubble Sort because:

a)Better Average Case Performance

* Quick Sort: O(n log n)
* Bubble Sort: O(n²)  
   Quick Sort is significantly faster for most inputs.

b)Efficient for Large Datasets

* Quick Sort handles large arrays efficiently.
* Bubble Sort becomes very slow as data size grows.

c)Widely Used in Real Systems

* Quick Sort (especially dual-pivot versions) is used in languages like Java (Arrays.sort()), Python (Timsort uses Quick Sort principles).

d)Less Number of Swaps

* Quick Sort typically performs fewer swaps than Bubble Sort, reducing overhead.

e)Divide and Conquer Approach

* Quick Sort splits the problem recursively, making it easier to parallelize and optimize.
* Bubble Sort is sequential and inefficient in logic